A Time-independent Deformer for Elastic Contacts

Camille Brunel, Pierre Bénard, Gaël Guennebaud

A Time-independent Deformer for Elastic Contacts

CAMILLE BRUNEL, Inria, Univ. Bordeaux, France
PIERRE BÉNARD, LaBRI (UMR 5800, CNRS, Univ. Bordeaux), France
GAËL GUENNEBAUD, Inria, Univ. Bordeaux, France

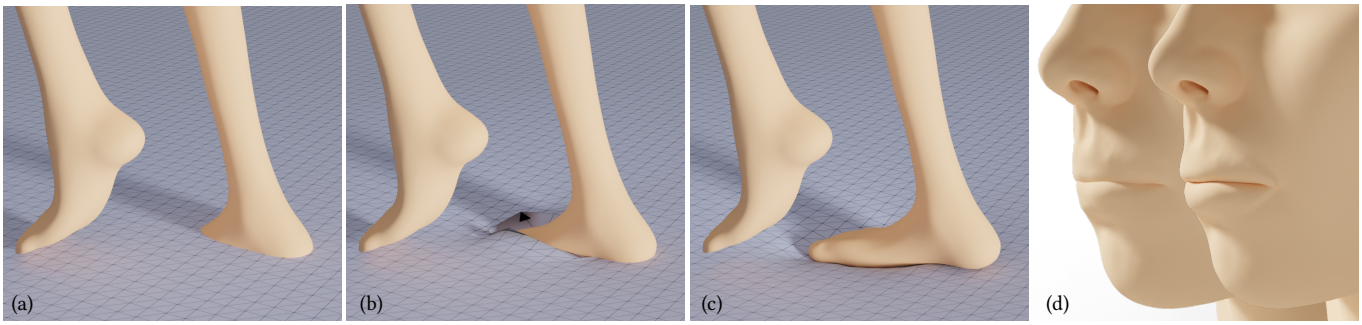


Fig. 1. (a) A rigid foot colliding with an elastic plane using (b) the approach of Brunel et al. [2020] compared to (c) our method. (d) In addition, our method handles elastic-elastic contacts such as in this example of a skinned mouth; the self-intersecting input configuration is shown in the background.

We present a purely geometric, time-independent deformer resolving local contacts between elastic objects, including self-collisions between adjacent parts of the same object that often occur in character skinning animation. Starting from multiple meshes in intersection, our deformer first computes the parts of the surfaces remaining in contact, and then applies a procedural displacement with volume preservation. Although our deformer processes each frame independently, it achieves temporally continuous deformations with artistic control of the bulge through few pseudo-stiffness parameters. The plausibility of the deformation is further enhanced by anisotropically spreading the volume-preserving bulge. The result is a robust, real-time deformer that can handle complex geometric configurations such as a ball squashed by a hand, colliding lips, bending fingers, etc.

CCS Concepts: • **Computing methodologies** → *Procedural animation*; *Mesh geometry models.*

Additional Key Words and Phrases: geometry processing, deformations, animation

ACM Reference Format:
Camille Brunel, Pierre Bénard, and Gaël Guennebaud. 2021. A Time-independent Deformer for Elastic Contacts. *ACM Trans. Graph.* 40, 4, Article 159 (August 2021), 14 pages. https://doi.org/10.1145/3450626.3459879

1 INTRODUCTION

Elastic objects are notably hard to animate by a CG artist, especially when they collide with each others, because reproducing their squashing and stretching behavior implies to manually craft plausible deformations in both space and time (e.g., using lattice deformers [Nieto and Susín 2013]). The standard solution to circumvent this problem is to rely on physical simulation [Nealen et al. 2006]. However such an approach only provides to the artist an indirect control of the elastic behavior, through physical parameters, which requires expertise and a time-consuming trial-and-error approach. Moreover, it does not easily allow the exaggeration of the deformation, which is commonplace in cartoon animation. In addition, due to their time-dependency, physical simulations must be run after the rigging and animation steps, preventing non-linear editing of the 3D scene. For character animation with contacts, quasi-static simulations (e.g., [McAdams et al. 2011]) and specific skinning techniques (e.g., [Vaillant et al. 2013]) partially lift this limitation, but they cannot be trivially extended to arbitrary elastic objects. Geometric modelling approaches handling collisions (e.g., [Li and Barbič 2019]) can be applied in this context, but they require expensive optimization and lack artistic controls of the deformation.

Recently, Brunel et al. [2020] have presented a geometric, time-independent approach to resolve local contacts between an elastic and a rigid object, producing plausible and art-directable bulge deformations. This method relies on the computation of two regions on the elastic object: the *contact* zone which corresponds to the part where the two objects in collision will remain in contact, and the *deformable* region that will be smoothly deformed to counterbalance the volume initially enclosed by the surfaces in intersection. The extent and shape of those regions are controlled by few, simple parameters, and the resulting deformation can be computed instantaneously at any frame of an animation.

However, generalizing this technique to two elastic objects in contact is very challenging because a contact zone and a deformable region need to be defined on both meshes, and most importantly, the contact zones must be consistent with each other since this region is shared by the two surfaces once the collision is resolved. In [Brunel et al. 2020], the contact zone is established by computing a bijection between the two surface regions within the intersection,

Authors' addresses: Camille Brunel, camille.brunel@inria.fr, Inria, Univ. Bordeaux, France; Pierre Bénard, pierre.benard@labri.fr, LaBRI (UMR 5800, CNRS, Univ. Bordeaux), France; Gaël Guennebaud, gael.guennebaud@inria.fr, Inria, Univ. Bordeaux, France.

and then extracting the subpart of the matched surfaces that should remain in contact via a ball-testing heuristic. Since this heuristic is highly asymmetric, it cannot be applied nor easily extended to elastic-elastic contacts. Moreover, the required bijection inevitably leads to strong mapping distortions and objectionable discretization artifacts for deep collisions. As observed but ignored by Brunel et al., the expected contact zones may also extend beyond the intersection region (Figure 1(a-c)).

To address these issues altogether, our core contribution is to compute a new partial matching between the two surfaces extending beyond the intersection region, and to use the resulting mapping to extract contact zones that are (1) consistent with each other, (2) stable during an animation without introducing any temporal dependencies, and (3) controllable by an artist with intuitive pseudo-stiffness parameters. To make this problem tractable in an interactive system, we make the key assumption that, at a given frame, the relative motion between the two objects is a pure translation. It implies that all the points within the contact zones share the same mapping direction, which allows us to efficiently compute *mapping regions* tightly enclosing the points that can belong to the contact zones. These points are then projected on a smooth potential contact surface, constructed according to the ratio of pseudo-stiffness parameters. A subset of them is selected to be part of the contact zones with a novel user-controlled symmetric criterion, ensuring C^1 continuity at its boundary. The deformation method of Brunel et al. [2020] can then be applied, with some improvements, independently on each mesh. The resulting deformation (Figure 1(c)) looks plausible and yet the bulge can be fully controlled by an artist. In addition, it is continuous in space, in time and when the relative stiffness changes.

Our second major contribution is to adapt this general pipeline to handle local self-collisions (Figure 1(d)). Our deformer can thus be used in the context of skeletal animation with skinning to automatically resolve collisions between adjacent body parts (e.g., arms, phalanxes, lips...). Last but not least, we present a new mechanism to anisotropically spread the amount of bulge in the deformation. Our heuristic takes into account the 3D spatial relationships between the contact surface where the forces are applied and the deformable surface to produce even more plausible deformation responses.

2 PREVIOUS WORK

Soft body simulation. Starting with Lasseter's discussion of squash and stretch in 3D animation [Lasseter 1987], and the seminal work of Terzopoulos et al. [1987] on elastic models, the quest for physically plausible deformations of soft bodies led to a large body of techniques, including local deformations during contacts such as the grasping task simulation of Gourret et al. [1989]. However most of these techniques fall into the category of physical simulation [Nealen et al. 2006]. This category encompasses many approaches (mass-spring systems, finite element methods, position-based dynamics, etc.) making different trade-offs between speed and accuracy, and the recent method of Li et al. [2020] even allows the user to specify a target accuracy while ensuring intersection- and inversion-free deformations. Nevertheless these methods share the two same major limitations: time-dependency and little intuitive artistic controls. These two constraints make simulation techniques difficult to use by animators while interacting with 3D objects.

Closer to our goals, the method of Pauly et al. [2004] focuses on local collisions by explicitly computing the contact surface between "quasi-rigid" objects and distributing the traction forces that act on their surfaces to drive a rigid-body simulation. The resulting deformation is volume preserving but limited to normal displacements and not art-directable.

Geometric deformations. A wide range of techniques have been proposed to deform surfaces: blend shapes, lattice and cage-based deformers [Nieto and Susín 2013; Sederberg and Parry 1986], discrete Laplace and Poisson models [Sorkine and Botsch 2009], etc. Yet, to the best of our knowledge, only a few methods ensure collision-free, volume-preserving deformations by design. Two field-based space-deformation techniques [Angelidis et al. 2006; von Funck et al. 2006] fall into this category, but the former cannot handle collisions between arbitrary objects, and the latter requires expensive numerical integration. The geometric framework of Harmon et al. [2011] resolves surface intersections interactively during geometric modeling. This method is independent of the deformation model, but involves a computationally extensive numerical constrained optimization and lacks artistic controls of the deformation. The method of Li and Barbič [2019] showed improved results for the specific case of handle-based As-Rigid-As-Possible (ARAP) deformation [Sorkine and Alexa 2007], but otherwise suffers from similar limitations.

Articulated characters. Skin deformations can be instantaneously obtained during skeleton animation [Kavan et al. 2008; Le and Hodgins 2016; Le and Lewis 2019; Magnenat-Thalmann et al. 1988], but neither contacts nor volume preservation are handled by these approaches, requiring additional manually sculpted pose-space deformations [Lewis et al. 2000]. Specific skinning techniques [Kavan and Sorkine 2012; Rohmer et al. 2009; von Funck et al. 2008] ensure local or global preservation of the volume, but still cannot respond to collisions. Kinodynamic skinning [Angelidis and Singh 2007] both preserves the skin volume and avoids surface self-intersections, but it requires temporal integration and offers limited, indirect artistic control through painted weights. Implicit skinning [Vaillant et al. 2013, 2014] can also handle local collisions of the skin between neighboring articulations without resorting to simulation thanks to its implicit volumetric representation of the character. Yet its gradient-based operators are difficult to art-direct, and distant contacts are too expensive to handle at interactive rates.

By coupling skinning and quasi-static simulations, hybrid approaches [Gao et al. 2014; Liu et al. 2013; McAdams et al. 2011; Smith et al. 2018; Teng et al. 2014] can respond to any kind of collisions in a time-independent fashion. However, these methods either involve costly iterative optimization only suitable for offline computation, or, for sub-space approaches, require manually generated training poses to accelerate the simulation. Using simpler position-based models [Bender et al. 2015; Bouaziz et al. 2014], local collisions can also be supported at interactive rates [Abu Rumman and Fratarcangeli 2015; Deul and Bender 2013; Komaritzan and Botsch 2018]. Even though these methods may help fixing skinning artifacts, such as local surface self-intersections near joins, they are not designed for distant contacts between body parts or

with external objects. In addition, they may suffer from stability issues. Recently, Zhang et al. [2020] have shown how elastodynamic secondary effects can be simulated on top of a rigged shape, in particular during collisions. It is complimentary to our method that focuses on instant deformations without dynamics.

3 METHOD OVERVIEW

In this work, we consider multiple elastic objects, or object parts in intersection. For the sake of simplicity, we will start in this overview with a pair of distinct objects (e.g., the blue plane and green sphere in Figure 2). In this case, our method follows the general pipeline introduced by Brunel et al. [2020] for elastic-rigid contacts, but we entirely revisit the contact surface computation (Section 4), as well as the computation of the displacement direction (Section 5.2).

To compute the contact surface, we first devise a new partial matching between the two surfaces extending beyond the intersection region. We compute such correspondences per vertex of each mesh (depicted with dotted lines in Figure 2(a)) by finding the most distant points on the other surface along a unique *mapping direction* $\hat{\mathbf{d}}$ defined in Section 4.1. This yields the so called *mapping regions* $\mathcal{M}_1$ and $\mathcal{M}_2$, represented in lighter colors in Figure 2(a). To avoid considering the entire meshes (e.g., by casting rays for all vertices), we describe in Section 4.2 an efficient algorithm based on the surface silhouettes seen from $\hat{\mathbf{d}}$ to conservatively restrict the computation of the mapping.

Then, as presented in Section 4.3, we determine a potential contact surface $\mathcal{S}$ (in grey in Figure 2(a)) as the linear interpolation of the mapped surfaces according to the ratio of their user-controlled pseudo-stiffness parameters k_1 and k_2. After projecting the mapped vertices of both meshes onto $\mathcal{S}$, we then determine the subset of these vertices that will belong to the final contact zone C (in red in Figure 2(b)) using a new user-controllable symmetric criterion detailed in Section 4.4.

Finally, we need to deform each mesh over a user-controlled geodesic distance around their contact zones in response to the collision (Figure 2(c)). Apart from some details presented in Section 5, we apply independently on both meshes the method of Brunel et al. that we briefly recall in the following. The deformable region of each initial mesh is displaced along a smooth unit vector field $\mathbf{d}$ whose amplitude is controlled by a 1D profile curve $\mathcal{H}_{a_i,s_i}$ instantiated at every vertex $\mathbf{p}_i$ of the mesh and evaluated using a one-dimensional radial parametrization $u_i \in [0,1]$, yielding to the final position:

$$\mathbf{p}'_i = \mathbf{p}_i + \mathcal{H}_{a_i,s_i}(u_i)\,\mathbf{d}_i. \tag{1}$$

The amplitude a_i and slope s_i of the profile curve at $u = 0$ are automatically computed to ensure C^1 continuity at the boundary of the contact zone, and interpolated by harmonic diffusion over the whole deformable region so that the previous equation can be evaluated everywhere. Beforehand, the parametrization u is computed using a variant of the heat-method [Crane et al. 2013], and the direction field $\mathbf{d}$ is computed using a new simpler, faster and smoother procedure (Section 5.2) ensuring that it smoothly varies from the mapping direction $\hat{\mathbf{d}}$ inside the contact zone to the normals of the initial mesh along the external boundary of the deformable region (inset in Figure 2(c)).

The profile curve $\mathcal{H}$ is implemented as a parametric cubic B-spline curve exhibiting one degree of freedom h_v controlling the bulge, and computed in such a way as to ensure exact volume preservation (Section 5.3). The bulge can be artistically controlled to exaggerate or cancel the volume compensation, and anisotropically spread over the deformable region to produce even more plausible deformation responses (Section 7). We provide the pseudo-code for the full algorithm in Appendix B.

Last but not least, the collision between adjacent parts of the same object, e.g., around an articulation in the context of skinning, is handled with some adjustments of the pipeline detailed in Section 6.

4 CONTACT SURFACE

Our general algorithm takes as input a set of open *working regions* $\{\mathcal{W}_l\}$, each $\mathcal{W}_l$ enclosing the expected deformable regions. The *contact surface* extraction procedure described in this section is carried out independently on each pair of colliding working regions. We will thus consider a single pair, with $l \in \{1,2\}$. On each of theses working regions, we compute the part of the mesh in intersection denoted $\mathcal{I}_1$ and $\mathcal{I}_2$. Those intersection regions are delimited by the set of edge-face intersection points.

For a given frame, our goal in this section is to compute the *contact surface* $\mathcal{S}$ between the two objects without using any information from previous frames. We aim to obtain two contact zones C_1, C_2 and two bijective mappings between them and the contact surface. Each vertex $\mathbf{p}_i \in C_l$ then has a corresponding position $\mathbf{p}'_i \in \mathcal{S}$, and the pair of points $(\mathbf{p}_i, \mathbf{p}'_i)$ defines a displacement direction $\mathbf{d}_i$. Since these ingredients are tightly coupled, computing them at once would require some non-linear optimization which seems both hazardous and prohibitively costly to achieve interactive performance. Instead, we simplify the problem by considering that the relative motion between the two objects is a pure translation in the direction $\hat{\mathbf{d}}$ (which can vary over the course of the animation) and by ignoring sliding within the contact region. As a result, all the vertices of a given contact zone share the same mapping direction towards the other mesh and the contact surface. This yields the following four steps strategy:

(1) compute the mapping direction $\hat{\mathbf{d}}$ (if not set by the artist),
(2) establish the mapping between the two meshes for supersets, called *mapping regions* $\mathcal{M}_l$, of the yet unknown contact zones,
(3) compute the corresponding contact surface $\mathcal{S}$,
(4) extract the contact zones C_l.

Each step is fully detailed bellow.

4.1 Mapping direction

To determine the direction $\hat{\mathbf{d}}$ at each frame of an animation, we propose an automatic method based on the geometry of the meshes in collision.

For each mesh l, we compute the mean normal $\bar{\mathbf{n}}_l$ over its intersection region $\mathcal{I}_l$. The idea is then to define the direction $\hat{\mathbf{d}}$ as a weighted combination of these two mean normals, giving a higher weight to the object having the smallest variation of normals within the intersection region. Using $1/\mathrm{Var}(\mathbf{n}_l)$, the inverse of the *total*

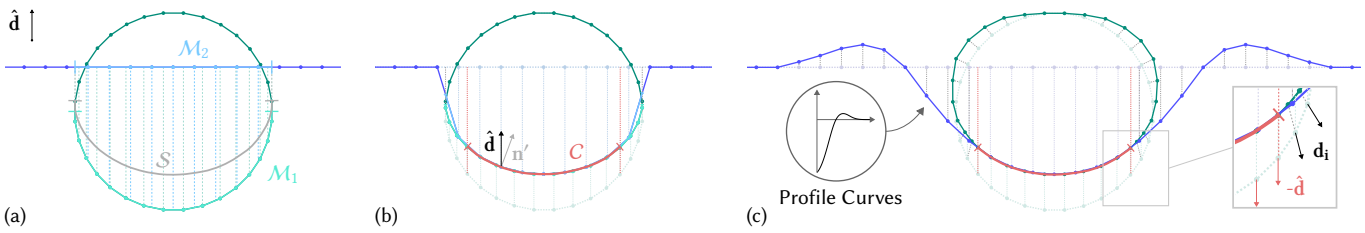


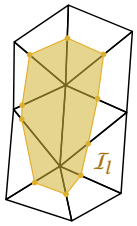
Fig. 2. **Overview of our approach.** At each frame independently, we consider two elastic objects and detect their intersection. (a) To resolve the collision, a matching between the two surfaces (dotted lines) is computed along a unique direction $\hat{\mathbf{d}}$ resulting in the definition of a mapping region $\mathcal{M}_{l\in\{1,2\}}$ (lighter color) on each mesh. A shared potential contact surface $\mathcal{S}$ (in grey) is determined using the stiffness ratio of the two objects. (b) Each mesh is then projected onto this intermediate surface along $\hat{\mathbf{d}}$, and a contact zone C (in red) is defined based on the geometry of $\mathcal{S}$, using the angle between $\hat{\mathbf{d}}$ and the normal $\mathbf{n}'$ on $\mathcal{S}$, and a user-controlled parameter. (c) The extent of the deformation and its displacement direction (dashed lines), defined by a unit vector field $\mathbf{d}_i$, are computed by multiple diffusions. The displacement magnitude is described by a family of profile curves automatically adjusted to ensure exact volume preservation. This displacement is eventually applied to the initial meshes (desaturated color) to obtain the final shape of each object.

variation (i.e., the trace of the covariance matrix), as weights yields:

$$\hat{\mathbf{d}} = \frac{-\bar{\mathbf{n}}_1 \mathrm{Var}(\mathbf{n}_2) + \bar{\mathbf{n}}_2 \mathrm{Var}(\mathbf{n}_1)}{\| -\bar{\mathbf{n}}_1 \mathrm{Var}(\mathbf{n}_2) + \bar{\mathbf{n}}_2 \mathrm{Var}(\mathbf{n}_1)\|} \tag{2}$$

Note that, since the normal fields of the two meshes are in opposite directions, a minus sign is required here. We arbitrarily chose to orient $\hat{\mathbf{d}}$ in the direction of the normal field of the second mesh.

To maintain a flowing animation, this direction $\hat{\mathbf{d}}$ must smoothly vary in time. Let us observe that the intersection region is continuous in time as it is delimited by the exact points of intersection between the two objects (see inset). The temporal continuity of $\hat{\mathbf{d}}$ can thus be obtained by computing the mean and total variation through continuous integrals of a continuous normal field. For the sake of simplicity, we consider the unnormalized vector field obtained through piecewise linear interpolation, allowing for simple Gauss quadrature integration. For instance, the mean $\bar{\mathbf{n}}_l$ is given by:



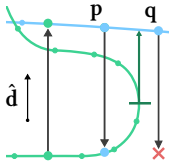
$$\bar{\mathbf{n}}_l = \frac{\mathbf{m}_l}{\|\mathbf{m}_l\|}, \quad \text{with} \quad \mathbf{m}_l = \sum_{f\in\mathcal{I}} A_f \mathbf{n}_f,$$

where A_f is the area of triangle face f (possibly coming from the cut of an initial triangle), and $\mathbf{n}_f$ denotes its average vertex normals. The same method, using a degree two Gauss quadrature rule, is applied to compute the total variation.

If the user is not satisfied with this automatically computed direction, she or he can specify it manually and even key-frame it during the animation. For example, this can be used to fake the influence of tangential motion and friction forces.

4.2 Mapping computation

We define the mapping regions $\mathcal{M}_l$ as the set of points (not only vertices) on each mesh that can be mapped to each other. To this end, each object is projected onto the other by matching every vertex with the most distant point on the opposite mesh, when it exists, along the associated mapping direction (inset figure). Note that these points (e.g., $\mathbf{p}$ in the inset) do not necessarily belong to the intersection zones; unlike [Brunel et al. 2020], our mapping regions $\mathcal{M}_l$ can thus be larger than $\mathcal{I}_l$. The simplest approach to compute this matching consists



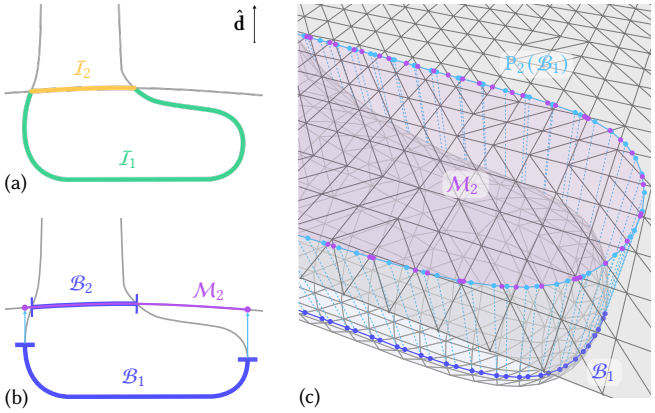


Fig. 3. **Mapping region definition** (a) The front-facing regions $\mathcal{B}_l$ are computed in the part of the objects in intersection $\mathcal{I}_l$ (in green for Mesh 1, yellow for Mesh 2). (b) The intersection between $\mathcal{B}_1$ and $\mathcal{I}_1$ (blue) is projected on $\mathcal{W}_2$ following $\hat{\mathbf{d}}$. The union of this projected region and $\mathcal{B}_2$ defines $\mathcal{M}_2$ (purple) (c) 3D View of (b). The chained contour of $\mathcal{B}_l$ (blue) is projected on Mesh 2. The intersection between the projected chaining (cyan) and the edges of Mesh 2 defines the exact-boundary point of $\mathcal{M}_2$ (purple points).

in casting a ray along $\hat{\mathbf{d}}$ (resp. $-\hat{\mathbf{d}}$) for each vertex in the working region $\mathcal{W}_l$ of each object and to find its furthest intersection with the opposite mesh. Such an approach would not only be inefficient, since most rays will not find any intersection, but more importantly, it will prevent us to tightly define the mapping regions. If these regions are not conservative, then temporal instabilities will occur when the contact zone reaches the boundary of the mapping region. In contrast, conservatively defining $\mathcal{M}_l$ as the set of faces containing at least one matched point (e.g., the segment $\mathbf{pq}$ in the inset figure) would make further processing (i.e., the construction of the contact surface and contact zones) impossible due to unmatched points (namely, $\mathbf{q}$ which is beyond the projection of the green mesh silhouette onto the blue surface). To solve both problems, we need an efficient algorithm to compute mapping regions which tightly enclose all the points that can be projected onto the opposite surface, and which smoothly evolve over mesh edges. This algorithm proceeds in two steps.

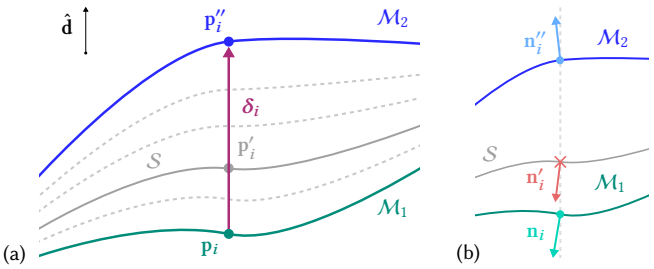


Fig. 4. **Potential contact surface definition.** (a) The shape of the potential contact surface $\mathcal{S}$ is a linear combination of $\mathcal{M}_1$ and $\mathcal{M}_2$ using $\bar{k}_l$. The displacement $\boldsymbol{\delta}_i$ of a point $\mathbf{p}_i$ belonging to $\mathcal{M}_1$ is a portion of its full projection $\tilde{\mathbf{d}}_i$ on $\mathcal{M}_2$ along $\hat{\mathbf{d}}$. The solid line corresponds to $\bar{k}_1 = 0.66$ and $\bar{k}_2 = 0.34$. (b) The normal $\mathbf{n}'_i$ of a vertex i on $\mathcal{S}$ is obtained by linear interpolation of the normal $\mathbf{n}_{i_1}$ and $\mathbf{n}_{i_2}$ of its associated points on $\mathcal{M}_1$ and $\mathcal{M}_2$ respectively.

First, we find the smallest region $\mathcal{B}_l$ that needs to be projected onto the opposite mesh to resolve the collision. Based on the observation that the parts of two colliding objects that remain in contact must face each other, $\mathcal{B}_l$ corresponds to the *front-facing* parts of the surfaces in intersection $\mathcal{I}_l$ when seen from their associated mapping direction, as illustrated in Figure 3(a-b). More precisely, defining the orientation function $g = \mathbf{n}^\top \hat{\mathbf{d}}$, a point on the surface is front-facing if $g \leq 0$. These regions are, by definition, delimited by the occluding contours of the surface. To ensure temporal continuity on polygonal meshes, we extract "interpolated" contours for all faces within $\mathcal{I}_l$ using the method of Hertzmann and Zorin [2000]. By definition, an edge for which the orientation function has opposite signs at its two vertices i and j is crossed by a contour. The barycentric position t of the contour point along this edge is obtained by linear interpolation of g, i.e., $t = g_i/(g_i - g_j)$. The resulting contour consists in line segments inside faces that smoothly delineate $\mathcal{B}_l$.

In a second step, we need to find the corresponding region on the opposite mesh. In the following we describe our method for the first object; it works similarly for the second one. To obtain continuous regions defined on mesh edges, we extrude the chained boundary segments of $\mathcal{B}_1$ (dark blue ticks and curve in Figure 3(b), resp. (c)) along $\hat{\mathbf{d}}$, and consider its intersection with the front-facing part of the other object $\mathcal{W}_2$ (light blue curve). Each intersection between an extruded quad and a front-facing edge of $\mathcal{W}_2$ (purple dots), i.e., such that $g(t) \leq 0$ with t the intersection parameter along the edge, delineates the boundary of the projected region $\mathrm{P}_2(\mathcal{B}_1)$. We tag the extremities of each intersected edge as either *inside* or *outside* $\mathrm{P}_2(\mathcal{B}_1)$ according to their relative positions with respect to the quad. The closed region $\mathrm{P}_2(\mathcal{B}_1)$ is then obtained by propagating the *inside* tags. Eventually, the final mapping region $\mathcal{M}_2$ is obtained as the union of $\mathcal{B}_2$ and $\mathrm{P}_2(\mathcal{B}_1)$; this is required in more complex configurations than the one depicted in Figure 3 for which $\mathcal{B}_2$ is already included in $\mathrm{P}_2(\mathcal{B}_1)$.

Once the mapping regions have been determined for both meshes, we project all the vertices inside them by casting rays along $\hat{\mathbf{d}}$ (resp. $-\hat{\mathbf{d}}$) and finding the most distant point on the opposite mesh, which is now guaranteed to exist.

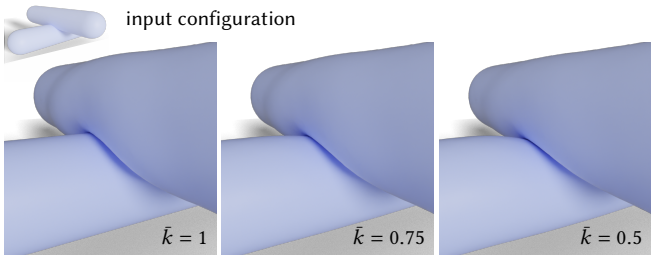


Fig. 5. **Variations of the relative stiffness parameter** $\bar{k}$ between the two capsules. Here $\bar{k}$ is the relative stiffness of the lower capsule.

4.3 Potential contact surface

Once the mapping regions have been defined for each object, we need to determine the shape of the shared contact surface $\mathcal{S}$ resulting from the collision. In the limit asymmetric case involving a rigid object, the contact surface is the rigid mesh. For two elastic objets, this is not as straightforward; the intermediate surface is located between the two initial meshes and depends on their *relative stiffness*, defined for mesh l as:

$$\bar{k}_l = \frac{k_l}{k_1 + k_2},\ l \in \{1, 2\},$$

where $k_{l\in\{1,2\}}$ are user-controlled pseudo-stiffness parameters. The effect of $\bar{k}$ is depicted in Figure 5. The full projection of one mapping region onto the other, as defined in the previous section, corresponds to the displacement of a fully elastic surface onto a rigid object, i.e., a zero relative stiffness. Conversely, if the mapping region is not modified, it corresponds to a relative stiffness of 1. For intermediate values, we use a fraction of this displacement proportional to the relative stiffness of the considered mesh. For example, if a point $\mathbf{p}_i$ of $\mathcal{M}_1$ is mapped to $\mathbf{p}''_i$ on $\mathcal{M}_2$ by a displacement vector $\boldsymbol{\delta}_i = \mathbf{p}''_i - \mathbf{p}_i$, its position on $\mathcal{S}$ will be $\mathbf{p}'_i = \mathbf{p}_i + (1 - \bar{k}_1)\boldsymbol{\delta}_i$ (see Figure 4(a)).

To determine the extent of the contact zone and ensure C^1 continuity at its boundary (as detailed in Section 4.4), we need the normal at each of these projected points. Independently projecting the polygonal surface of $\mathcal{M}_1$ and $\mathcal{M}_2$ onto $\mathcal{S}$ and recomputing vertex normals would require to explicitly cut each mesh along their mapping regions boundary, and more importantly, it would lead to discretization errors that would break the symmetry of the contact zone extraction as it is based on the normal field of the contact surface (see next section). Instead, for every vertex i of $\mathcal{M}_1$, we show in Appendix A that its normal on the shared potential surface $\mathcal{S}$ can be directly computed from its initial normal $\mathbf{n}_i$ and the interpolated normal $\mathbf{n}''_i$ of its corresponding point on $\mathcal{M}_2$, when $\mathcal{S}$ is defined using the above linear interpolation (see Figure 4(b)). This yields the following formula:

$$\mathbf{n}'_i = \eta_i / \|\eta_i\|, \quad \text{with:}$$
$$\eta_i = (I - \hat{\mathbf{d}}^\top \hat{\mathbf{d}}) \left(\frac{\bar{k}_2\, |\hat{\mathbf{d}}^\top \mathbf{n}''_i|}{\mathrm{sign}(\hat{\mathbf{d}}^\top \mathbf{n}_i)}\, \mathbf{n}_i + \frac{(1 - \bar{k}_2)\, |\hat{\mathbf{d}}^\top \mathbf{n}_i|}{\mathrm{sign}(\hat{\mathbf{d}}^\top \mathbf{n}''_i)}\, \mathbf{n}''_i \right)$$
$$- |\hat{\mathbf{d}}^\top \mathbf{n}_i|\, |\hat{\mathbf{d}}^\top \mathbf{n}''_i|\, \hat{\mathbf{d}}\,.$$

To obtain the normal of vertex j of $\mathcal{M}_2$ on $\mathcal{S}$, we use $-\hat{\mathbf{d}}$ instead of $\hat{\mathbf{d}}$ in this equation.

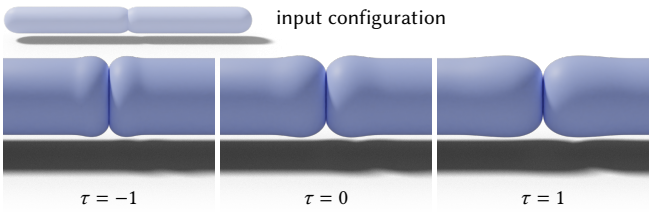


Fig. 6. **Variations of the apparent stiffness parameter** τ between two capsules, which controls the extent of the contact zone. To produce plausible results, the extent of the deformation area has been increased accordingly.

4.4 Contact zones

We observed empirically that the extent of the contact zone between two elastic objects in collision varies according to their respective geometry, stiffness and the depth of interpenetration. In the following, we define a criterion that captures these variations while respecting our application constraints, i.e., symmetry with respect to the two objects, temporal continuity, and artistic controllability.

To obtain a symmetric behavior, this criterion is based on the geometry of the common surface $\mathcal{S}$, and motivated by two empirical observations: (1) the more distant the normal of this surface is from the direction of collision, the less the two objects should remain in contact; (2) the deeper the collision is, the larger the contact zone should be. Therefore, we define our criterion as follows. On each mapping region $\mathcal{M}_l$, a vertex remains on the contact surface $\mathcal{S}$, that is, in the contact zone C_l, if it fulfills the condition:

$$\boldsymbol{\delta}_i^\top \mathbf{n}_i' \geq \varepsilon_c, \tag{3}$$

where, as before, $\mathbf{n}_i'$ is the normal of vertex i on $\mathcal{S}$, and $\boldsymbol{\delta}_i$ is the displacement between the vertex position and its projection on the other mesh. Since the average magnitude of those scalar products largely varies over the animation, the threshold ε_c needs to be continuously recomputed. To ensure a stable control during the animation, our idea is to start from the mean $m_\mathcal{S}$ of $\boldsymbol{\delta}_i^\top \mathbf{n}_i'$ over $\mathcal{S}$, and then deviate from this reference by a user-controlled amount τ of the standard deviation $\sigma_\mathcal{S}$ of $\boldsymbol{\delta}_i^\top \mathbf{n}_i'$ over $\mathcal{S}$:

$$\varepsilon_c = m_\mathcal{S} + \tau \sigma_\mathcal{S}. \tag{4}$$

As depicted in Figure 6, the scale-independent parameter τ allows the user to adjust the apparent stiffness of the most elastic object by controlling the extent of the contact zone, while its shape is determined by the relative stiffness $\bar{k}_l$ defining $\mathcal{S}$.

Moreover, to avoid too high tangential displacements in the final deformation, we add to Equation 3 the following safeguard symmetric condition on the angle between the initial surface normals and the mapping direction:

$$\min(-\hat{\mathbf{d}}^\top \mathbf{n}_i, \hat{\mathbf{d}}^\top \mathbf{n}_i'') \geq cos(80^\circ),$$

where $\mathbf{n}_i''$ denotes as before the normal of the point corresponding to the vertex i on the opposite object (see Figure 4).

With such a definition, the contact zone is discretized at the vertices of both meshes. Once more, we need a smooth temporal evolution of its boundary to guarantee a continuous deformation during animation, and we thus compute the exact-boundary points of the contact zone on mesh edges. For each outgoing edge ij joining a vertex i inside the contact zone and a vertex j outside of it, we find the smallest barycentric coordinate α at which the following conditions are met:

$$\left.\begin{aligned} \mathbf{n}'(\alpha)^\top \boldsymbol{\delta}(\alpha) &\geq \varepsilon_c \\ \min(-\mathbf{n}(\alpha)_1^\top \hat{\mathbf{d}}, \mathbf{n}(\alpha)_2^\top \hat{\mathbf{d}}) &\geq cos(80^\circ) \end{aligned}\right\} \tag{5}$$

where $\mathbf{n}'(\alpha)$ is the normal of that point on $\mathcal{S}$, $\boldsymbol{\delta}(\alpha)$ is its displacement, and $\mathbf{n}(\alpha)_{l\in\{1,2\}}$ are its initial normals on each mesh; all those quantities are independently obtained by linear interpolation of their respective values at the edge extremities. Once the minimum α value has be obtained by the resolution of a second degree equation, we project the corresponding interpolated point onto $\mathcal{S}$ following $\hat{\mathbf{d}}$ (or $-\hat{\mathbf{d}}$) and recompute its exact final position and normal that will be used in the subsequent steps of the pipeline.

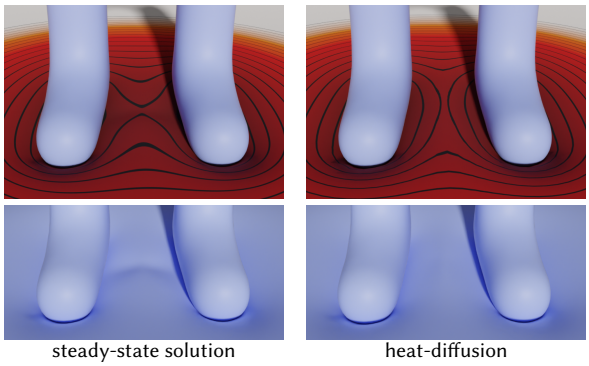


Fig. 7. **1D parametrization comparison.** A more geodesic-like parametrization (right) avoids unpleasant distortions when multiple contact zones are involved in the collisions.

5 FINAL DEFORMATION

Once the contact zones have been established for all pairs of colliding working regions, each of them is processed independently by computing:

(1) the extent of the deformation together with a 1D parametrization of the corresponding deformable region,
(2) a directional field supporting the vertex displacement,
(3) the final deformation of the surface along this field.

These steps are described below and summed up in Algorithm Part 2.

5.1 1D parametrization

As detailed by Brunel et al. [2020], the 1D parametrization of the deformable region, denoted u, ranges from 0 at the contact region boundary ∂C to 1 at the exterior boundary of the deformable region. In a nutshell, it is obtained by computing a smooth distance field ϕ to ∂C using the heat-method [Crane et al. 2013], which involves two partial first-order differential equations. One to compute a scalar field v through heat diffusion, for which we used a time step equals to $10\,h^2$, with h the average edge length in the current working region. And a second Poisson equation $\Delta\phi = -\nabla \cdot (\nabla v / \|\nabla v\|)$. Since the source ∂C is defined as an arbitrary polyline over the mesh, both problems are solved using linear constraints along edges and adequately weighted cotangent coefficients. Compared to Brunel et al. who used the steady-state solution to compute v, this results in more

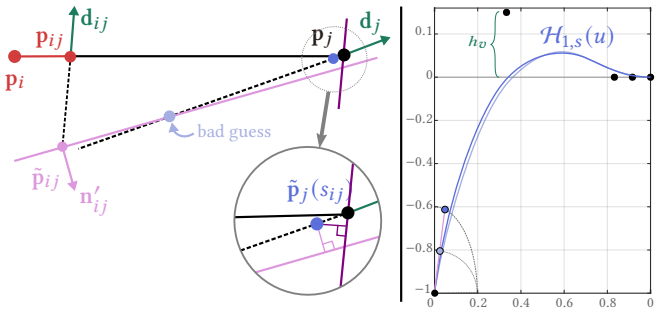


Fig. 8. **Slope.** Left: the profile slope at $u = 0$ for the vertex $\mathbf{p}_j$ and edge $i - j$ was initially estimated such that its expected final position $\tilde{\mathbf{p}}_j$ lies on the pink tangent plane. Constraining $\tilde{\mathbf{p}}_j$ not to move far away from the safeguard purple plane aligned with $\mathbf{d}_{ij}$ avoids the production of prohibitively large displacements in such extreme configurations for which the initial point is already close to the target tangent plane. Right: illustration of the profile curve $\mathcal{H}$ and its six control points. Adjusting the second one along an ellipse instead of a circle as in [Brunel et al. 2020], enforces the actual profile to stay closer to its tangent (purple line) around $u = 0$.

geodesic-like parameterization, and thus smoother deformation, as illustrated in Figure 7.

5.2 Direction field

The direction field $\mathbf{d}$ is expected to reproduce the known displacement directions along the contact zone boundaries at $u = 0$ and to smoothly vary until the surface normals are reached at $u = 1$. To this end, Brunel et al. performed a direct harmonic interpolation of these constraints through costly parallel transport of the tangential component of the directions.

Instead, we propose a computationally efficient two step process. We first compute a mapping vector field $\bar{\mathbf{d}}$ over the working region through a standard harmonic diffusion of the mapping directions $\hat{\mathbf{d}}_l$ defined for each contact zone C_l resulting from the collision of the current working region with the other working regions $\mathcal{W}_l$ (Section 4.1). More precisely, we set linear constraints of the form $\bar{\mathbf{d}} = \hat{\mathbf{d}}_l$ along each contact zone boundary ∂C_l, and natural Neumann conditions over the remaining exterior boundary. In practice, this problem can be solved reusing the matrix factorized for solving the second Poisson problem of the previous modified heat-method, which makes this diffusion inexpensive. Moreover, this step can be by-passed in the usual case of a single object colliding with the current working region, since the result is a constant field $\bar{\mathbf{d}} = \hat{\mathbf{d}}$.

In a second step, this vector field is blended with the surface normal field using the 1D parametrization u as weights to accomplish the desired transition:

$$\mathbf{d}_i = \text{normalize}\left(w(u_i)\,\bar{\mathbf{d}}_i + (1 - w(u_i))\,\mathbf{n}_i\right), \tag{6}$$

with $w(u) = (1 - u^2)^6$. This weighting function enables an asymmetric and rapid transition toward the normal field.

5.3 Final steps

The final steps of the deformation are carried out as in the original method of Brunel et al. [2020] with few implementation changes that we briefly mention here.

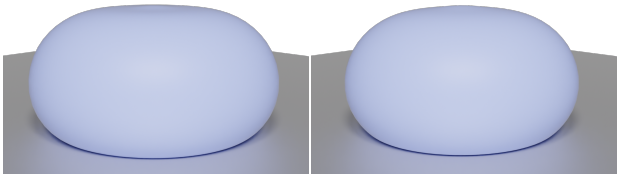


Fig. 9. **Comparison of volume preservation.** Left: Linear approximation [Brunel et al. 2020]. Right: Our exact method.

Parameter diffusions. Let us recall that the profile curve $\mathcal{H}$ is parametrized by its *amplitude* a, and *slope* s at $u = 0$ such that: $\mathcal{H}_{a_i,s_i}(0) = a_i$, and $\mathcal{H}'_{a_i,s_i}(0) = s_i$. Those parameters are first estimated for every vertices adjacent to the contact zones, and those values are then diffused over the rest of the deformable region by harmonic diffusion. In our implementation, we use natural Neumann conditions at the exterior boundary instead of average Dirichlet boundary conditions. This allows for a better preservation of the boundary values along the gradients of u.

Slope estimation. Prior to diffusion, to estimate the slope s_{ij} at a vertex j of and edge $i - j$ crossing the contact zone boundary, Brunel et al. used a linear approximation of the final displacement:

$$\tilde{\mathbf{p}}_j(s_{ij}) = \mathbf{p}_j + (-a_j + u_j\, s_{ij})\,\mathbf{d}_j\,,$$

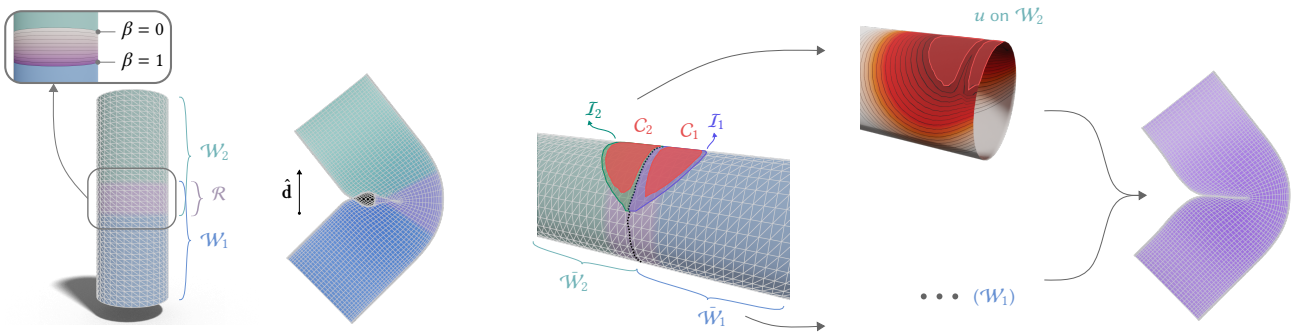
and then estimated s_{ij} such that $\tilde{\mathbf{p}}_j$ lies on the tangent plane with normal $\mathbf{n}'_{ij}$ of the nearby contact surface (pink line in Figure 8-left):

$$(\tilde{\mathbf{p}}_j(s_{ij}) - \tilde{\mathbf{p}}_{ij})^\top \mathbf{n}'_{ij} = 0 \quad \text{with} \quad \tilde{\mathbf{p}}_{ij} = \mathbf{p}_{ij} + a_j \mathbf{d}_{ij}.$$

However, as depicted in Figure 8-left, this problem becomes hill posed when the direction $\mathbf{d}_j$ is aligned with the edge $i - j$. We mitigate this shortcoming by constraining $\tilde{\mathbf{p}}_j$ to lie on the plane which is the most orthogonal to the edge and passing through $\mathbf{p}_j$ and $\mathbf{d}_{ij}$ (purple line), while solving for these two distance-to-plane equations in a least-square sense (blue point).

Moreover, we found that, when the slope is expected to be very steep (nearly vertical), the difference between this linear approximation and the actual displacement magnitude can be very large even if u_j is very small. We alleviate this issue by increasing the length of the tangent vector of the B-spline parametric curve according to the magnitude of the slope by adjusting the position of the second control point along an ellipse instead of a circle (see Figure 8-right). This way, the actual profile curve better respects the linear approximation made to estimate the slopes.

Volume preservation. To ensure pseudo-volume preservation, Brunel et al. linearly approximated the signed volume enclosed by the initial and displaced meshes, which can lead to overcompensation issues in some extreme cases, as seen in Figure 9. Our implementation considers an exact volume formula obtained by decomposing this volume into a set of tetrahedrons. When solving for the degree of freedom h_v in the profile curve $\mathcal{H}$, this yields a cubic equation, whose smallest positive root is the expected solution. In practice, this modification has little impact on the overall runtime performance.



(a) Rest pose (b) Input configuration (c) Contact zones (d) Parameter diffusion (e) Result

Fig. 10. **Adjacent region overview.** (a) Two overlapping regions entails a *shared region* $\mathcal{R}$ onto which partition of unity weights β are computed in a preprocess. (b) Input frame clipped to reveal the self-intersection. (c) The shared region is partitioned through an automatically computed cut (black dotted line), yielding to the non overlapping working regions $\bar{\mathcal{W}}_{l\in\{1,2\}}$ within which the contact zones are identified. (d) A deformation is then computed independently for each whole working region $\mathcal{W}_l$, starting with the diffusion of its parameters; the 1D parametrization u is shown here. (e) The final result is obtained by blending the two intermediate deformations over the shared region $\mathcal{R}$; a clipped view is used again to reveal the resolved contact and volume preserving bulge.

6 ADJACENT WORKING REGIONS

The pipeline described so far assumes that the working regions are clearly separated, with no overlap, and zero deformation at their external boundaries. Such assumptions, however, do not always hold. This is typically the case of a skinned articulation for which two adjacent parts are colliding with each other (e.g., elbow, knee, etc.). The deformation response should not vanish between them and, as such, it is not even possible to define a clear frontier separating these two parts. Another example of such a configuration are the lips of a closed mouth (Figure 1(d)).

We address this limitation by allowing adjacent working regions to overlap each other, while ensuring spatial continuity through a partition of unity blending of their respective deformation responses over the so called *shared region* $\mathcal{R}$ (depicted in purple in Figure 10(a)). Let β be the weights forming such a partition of unity, then the final position $\mathbf{p}'_i$ of each point in $\mathcal{R}$ is obtained as:

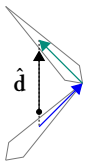
$$\mathbf{p}'_i = \beta_i\, \mathbf{p}'_{i,1} + (1-\beta_i)\, \mathbf{p}'_{i,2}\,,$$

where $\mathbf{p}'_{i,l}$ denotes the displaced position obtained by applying our pipeline to the working region $\mathcal{W}_l$.

This approach is very simple but requires additional special treatments. Most importantly, we have to automatically compute a strict partitioning of the working regions along the temporally-varying crease so that the computation of the contact surfaces (Sections 4.2 to 4.4) can be carried out. We also need to introduce some synchronization points during the computation of the deformation parameters of each working region such that the intermediate positions $\mathbf{p}'_{i,l}$ are already as close as possible to each other prior to averaging them. Those changes are detailed in the following paragraphs.

Shared region & Partition of unity. The shared region $\mathcal{R}$ is defined as the set of vertices that belong to the two user-defined working regions $\mathcal{W}_1$ and $\mathcal{W}_2$. It must be large enough to always include the crease appearing when the articulation bends, and designed in a symmetric fashion such that the expected crease occurs roughly at the middle of $\mathcal{R}$. The partition of unity weights β are precomputed on the rest pose of the skeleton by harmonic diffusion with Dirichlet boundary conditions such that $\beta = 1$ at the boundary vertices of $\mathcal{R}$ belonging only to the boundary of $\mathcal{W}_2$, and $\beta = 0$ for those belonging only to the boundary of $\mathcal{W}_1$ (Figure 10(a)). If $\mathcal{R}$ presents other boundaries, i.e.,vertices belonging to both working region boundaries, we use natural Neumann boundary condition for them.

Mapping direction. The heuristic described in Section 4.1 to compute the mapping direction does not apply in the current setting, because it requires strictly separated intersection regions. In addition, as detailed in the next paragraph, this mapping direction $\hat{\mathbf{d}}$ must partition the common intersection region. To work around this chicken-and-egg problem, it is usually possible to estimate $\hat{\mathbf{d}}$ from the relative motion between the two parts and, in some cases, it is even possible to instantaneously estimate it from the skeleton. For instance, in the common case of two bones connected at a joint (see inset), we compute $\hat{\mathbf{d}}$ as the vector orthogonal to the bisector of the bones lying within their supporting plane, which seems the natural direction to resolve the intersection within the fold.



Crease detection & Partitioning. In the typical scenario considered in this section, the intersection region of each working region is expected to be open with a common portion lying in the shared region (Figure 10(b-c)). To be compatible with the rest of our pipeline, this shared intersection region needs to be explicitly partitioned (between lines 1-2 of Algorithm Part 1). Intuitively, this partition cut should follow the crease produced by the input deformation (e.g., the skinning). This cut is expected to slightly vary throughout the animation, and must thus be computed at runtime. We identify this cutting crease using a graph-cut optimization [Boykov and Kolmogorov 2001] over the shared region driven by two heuristics. First, since the two intersection regions should face each other along the mapping direction $\hat{\mathbf{d}}$, the cut is expected to lie along the silhouette of $\mathcal{R}$ as seen from $\hat{\mathbf{d}}$. In practice, this is implemented as edge-weights w_e of the form:

$$w_e = 0.1 + \min_{\mathbf{x}\in e} |\mathbf{n}(\mathbf{x}) \cdot \hat{\mathbf{d}}|\,,$$

where $\mathbf{n}(\mathbf{x})$ denotes the surface normal at the point $\mathbf{x}$ within the edge e. Since $\mathbf{n}$ is obtained by linear interpolation along an edge,

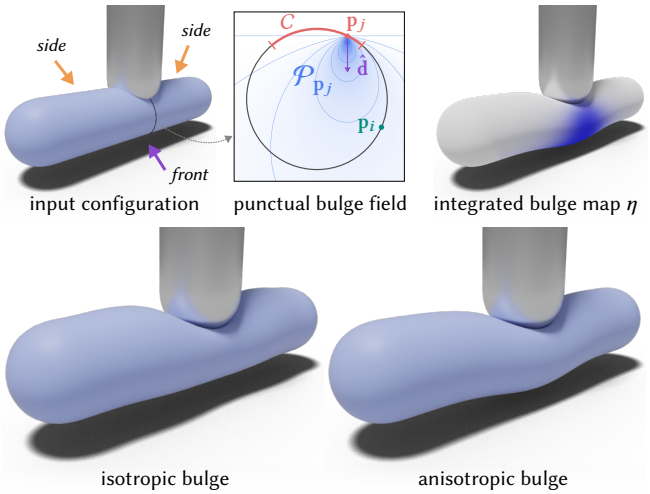


Fig. 11. **Anisotropic bulge.** Each vertex $\mathbf{p}_j$ of the contact zone C emits a punctual volumetric bulge field $\mathcal{P}_{\mathbf{p}_j}$ which is integrated for every point $\mathbf{p}_i$ of the deformable region to produce the bulge map η. This map is then used to anisotropically redistribute the volume of the deformation. [$\bar{k} = 1$, $\tau = 0$]

computing this minimum is straightforward: it is either 0 if a silhouette crosses the edge, or it is reached at one edge extremity. The second heuristic aims to regularize the previous criterion by assuming that the cut should most likely pass nearby the iso-contour $\beta = 0.5$ of the shared region. Within a graph-cut framework, this is easily implemented by using β as node-weights.

As depicted in Figure 10(c), this results in the segmentation of the shared region vertices into two parts, each of them forming a new truncated working region $\bar{\mathcal{W}}_l$.

Consistent contact zones. The computation of the contact surfaces (Sections 4.2 to 4.4) is carried out on these truncated working regions without any modification. We just need to ensure that the identified contact zones are properly reported on the parts of the opposite working region that have been temporarily cut out. For instance, in Figure 10(c-d) the region C_1 identified on $\bar{\mathcal{W}}_1$ is reported to $\mathcal{W}_2$.

Consistent mapping direction. Finally, we need to update the intermediate mapping direction field $\bar{\mathbf{d}}$ such that it smoothly transitions from $\bar{\mathbf{d}}$ to the opposite direction $-\bar{\mathbf{d}}$ over the shared region. This is accomplished using a linear interpolation parametrized by β:

$$\bar{\mathbf{d}} \leftarrow (2\beta - 1)\,\bar{\mathbf{d}}\,.$$

Implementation-wise, this could be done at line 43 of Algorithm Part 2. Note that the unnormalized vector field $\bar{\mathbf{d}}$ then vanishes at $\beta = 0.5$, and the final displacement direction $\mathbf{d}$ will thus gently fallback to the normal field according to Equation 6, which is the expected behavior (e.g., consider the exterior of an elbow).

7 ANISOTROPIC BULGE

So far, the bulge is isotropically spread around the contact zone, which might not always be desirable. To spatially control the amount of bulge, a simple approach consists in letting the user paint a so called *bulge map* η introduced in the profile curve $\mathcal{H}$ as a multiplicative factor of the ordinate h_v. By taking into account this new factor during the optimization of h_v (Section 5.3), the volume can still be globally preserved. A wide range of artistic controls is offered by allowing η to be negative, as in Figure 17-left where a very simple painting reveal fabric-like deformations.

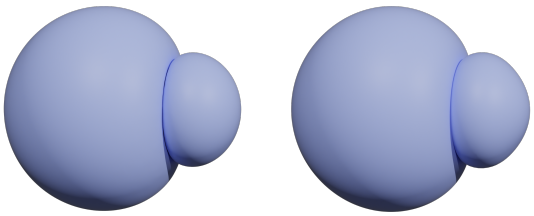


Fig. 12. **Comparison with a simulation.** Left: FEM simulation in Houdini©SideFX. Right: our method. [$\bar{k}_{\text{big}} = 0.6$, $\bar{\phi}_{\text{big}} = 250$, $\bar{\phi}_{\text{small}} = 150$, $\gamma = 0.9$, $\tau = 0.2$]

Such a map can also be procedurally generated to automatically produce plausible effects. In particular, as illustrated in Figure 11, it is often desirable to locate the bulge in *front* of the collision (purple arrow) rather than on its *sides* (orange arrows). As depicted in Figure 11 (top-middle), we mimic this intuitive behavior by attaching to every vertex $\mathbf{p}_j$ of the contact zone C a volumetric scalar field:

$$\mathcal{P}_{\mathbf{p}_j}(\mathbf{x}) = \frac{1}{\|\mathbf{x} - \mathbf{p}_j\|} \max\left(0, \frac{\hat{\mathbf{d}} \cdot (\mathbf{x} - \mathbf{p}_j)}{\|\mathbf{x} - \mathbf{p}_j\|}\right)^{\omega}$$

aligned with the mapping direction $\hat{\mathbf{d}}$ and decreasing according to both its distance to $\mathbf{p}_j$ and the cosine of the angle made between $\mathbf{x} - \mathbf{p}_j$ and $\hat{\mathbf{d}}$. This field vanishes for the points $\mathbf{x}$ lying on the plane with normal $\hat{\mathbf{d}}$ passing through $\mathbf{p}_j$. The exponent ω allows the user to balance the influence of the positional and directional terms ($\omega = 2$ in all our tests, unless specified otherwise). The bulge weight η_i of a point $\mathbf{p}_i$ of the deformation area is then obtained by summing up the contribution of each vertex in the contact zone C weighted by its penetration depth and associated area A_j:

$$\eta_i = \sum_{\mathbf{p}_j \in C} A_j \|\mathbf{p}'_j - \mathbf{p}_j\| \mathcal{P}_{\mathbf{p}_j}(\mathbf{p}_i)\,, \tag{7}$$

with A_j one third of the sum of the triangle areas adjacent to $\mathbf{p}_j$. When the collision produces multiple contact zones, this sum is computed over all of them, using the associated mapping direction to evaluate $\mathcal{P}_{\mathbf{p}_j}$.

8 RESULTS

Our method requires a few user inputs, namely the extent of the deformation region $\bar{\phi}$, the apparent stiffness parameter τ, and pseudo stiffness ratio $\bar{k}$. In the following, the later will be reported for the most deformable object giving its name as subscript. For instance, for a X-Y pair, $\bar{k}_{\text{X}} = 0.7$ means that the object X is the softer: it will go through 70% of the displacement while the other object Y will move by 30% only. Their respective effects on the deformation are visible in Figures 5 and 6. Since the value of $\bar{\phi}$ is scale dependent, it will be reported only when making sense. The other artistic parameters such as the shape of the profile curve $\mathcal{H}$, the mapping direction $\hat{\mathbf{d}}$, the volume conservation/exaggeration parameter γ, and the bulge map η will be reported only if they differ from their default settings.

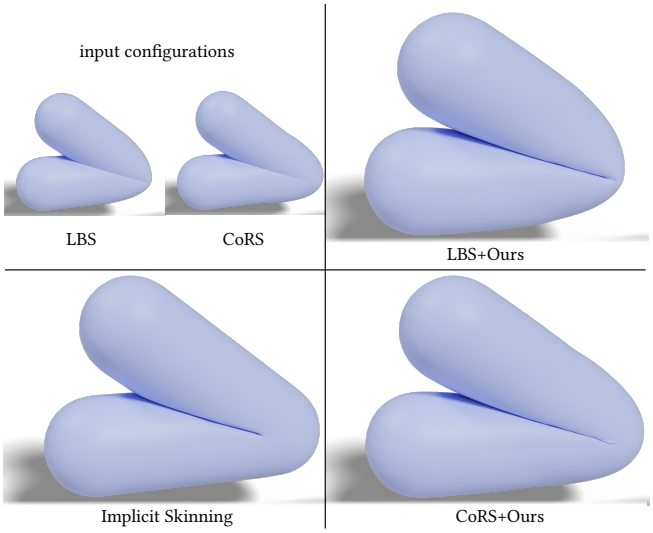


Fig. 13. **Comparison to Implicit Skinning** [Vaillant et al. 2014] on the bending of a skinned capsule user either Liner Blend Skinning (LBS) or Center of Rotation Skinning (CoRS) as input to our deformer. Notice the volume preserving bulge produced by our deformer. [$\bar{k} = 0.5$, $\tau = -1$]

All the results are provided as full animations in the accompanying video. The video also includes an interactive session demonstrating free manipulation of the objects and parameter adjustments.

8.1 Comparisons

Comparison to [Brunel et al. 2020]. Figure 1 compares our pipeline to the original approach of Brunel et al. in a case for which the latter is applicable, that is with one object being rigid, here the feet. As already noticed by the authors, their method struggles and eventually breaks when there is a too large distortion between the two intersected surfaces, such as when the whole foot goes below the ground. In contrast, our new techniques to match the surfaces and extract the contact zone are both agnostic to the actual intersection regions, which allows us to properly match the sole of the foot with its complete footprint on the ground.

Nevertheless, to ensure time-independence, we inherit one limitation of Brunel et al.: if one object entirely crosses the surface of the other, our method cannot generate any response. A practical workaround would be to ask the artist to provide a proxy geometry to recover missed intersections for very deep collisions.

Physical simulation. In Figure 12, we compare our method to a finite element simulation in Houdini© SideFX on two colliding elastic balls. Although our approach is designed to resolve local contacts, it manages to produce a very similar deformation. As can be seen in the video, however, compared to the time-dependent simulation, our result lacks dynamic inertia effects.

Comparison to Implicit Skinning. Figure 13 compares our deformer to the time-dependent Implicit Skinning technique [Vaillant et al. 2014]. Both techniques properly resolve the contact, but our deformer produces an additional volume preserving bulge.

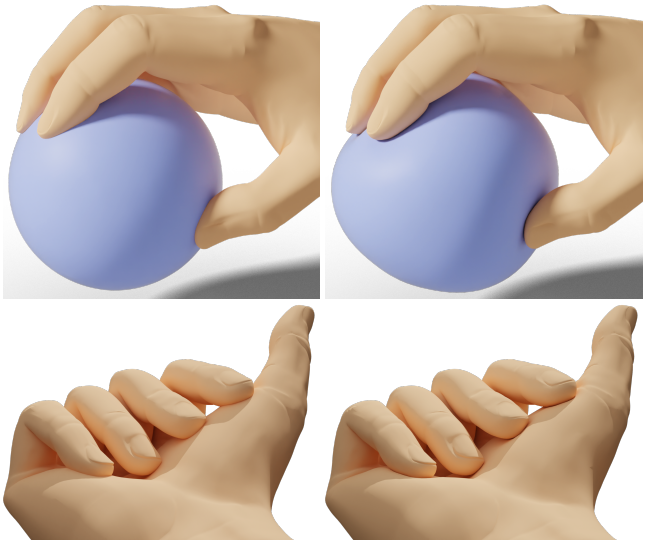


Fig. 14. **Multiple disconnected components.** Left column: input configurations, in both cases each finger produces an independent working region. Right: resulting deformations. Notice the subtle bulge of the finger tips (e.g., the thumb on the ball) and palm (e.g., around the little finger and between the index and middle fingers). [$\bar{k}_{\text{fingers}} = 0.8$, $\tau_{\text{ball}} = 0.6$, $\tau_{\text{palm}} = 0$, $\bar{\phi}_{\text{palm}} = \bar{\phi}_{\text{fingers}} = 1$, $\bar{\phi}_{\text{ball}} = 3$, $\gamma_{\text{ball}} = 3$, $\omega = 0$]

8.2 Qualitative evaluation

Multiple disconnected components. Figure 14 shows two examples with multiple soft components, the finger tips squashing a unique elastic ball or pressing the palm of its own hand. In these examples, the whole ball or the palm constitutes a single working region. As explained in Section 5, the contact zones are established independently for each finger tips, thus allowing each contact to be established with its own mapping direction and extent threshold ε_c, both being automatically computed.

Skinning & Self-intersections. Figure 15 illustrates the effects of our extended pipeline (Section 6) in the case of self-intersections occurring at the joint of an articulation with linear blend skinning. A slightly more complex example is shown in Figure 1(d) with self-intersections occurring between the two lips of a closed mouth animated using linear blend skinning. Whereas our automatic mapping direction heuristic works perfectly well for the finger, it does not apply to the mouth which is not an articulation. We thus manually set the mapping direction to a constant vertical vector.

Surfaces with complex reliefs. In Figure 16, we stress the robustness of our algorithms to the case of surfaces exhibiting relief details. This result has been obtained as is, with an automatically computed mapping direction (Section 4.1). As the motion of the jelly is rather tangential to the surface of the Bunny, the contact response could be improved by exploiting the relative motion of the two objects or keyframing the mapping direction. Besides, for surfaces exhibiting even more intricate details, it may be wise to precompute a smoothed normal field to drive the displacement directions in Equation 6.

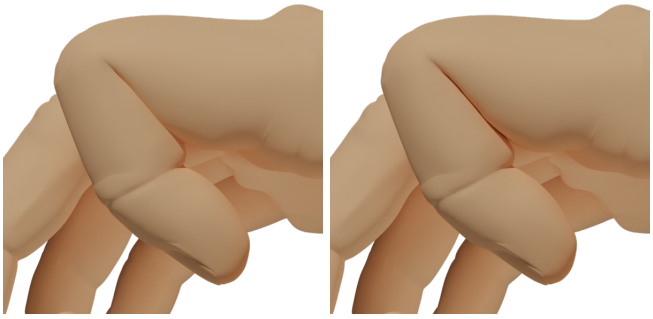


Fig. 15. **Skeletal skinning.** Left: input configuration. The shared region is between the two phalanxes. Right: resulting deformation. [$\bar{k} = 0.5, \tau = -1$]

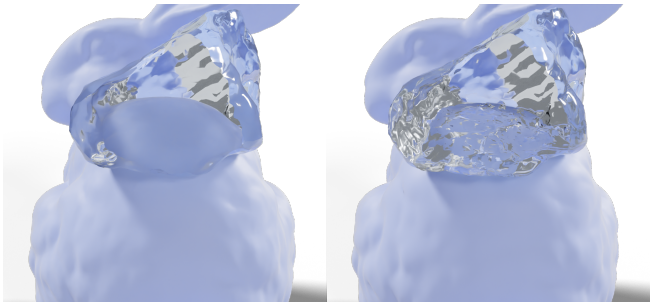


Fig. 16. **Surfaces with complex reliefs.** Left: input configuration. Right: resulting deformation. The jelly is clipped to show the resolved collision. [$\bar{k}_{\text{jelly}} = 0.6$, $\tau = -1$, $\bar{\phi}_{\text{bunny}} = 300$, $\bar{\phi}_{\text{jelly}} = 560$]

Artistic controls. Besides time-independence, artistic control is a central feature of our approach. For instance, Figure 17-left shows anisotropic *star-shaped* bulges achieved through a painted bulge map (Section 7). Moreover, our approach retains all the editing capabilities offered by the method of Brunel et al. [2020], including profile-curve tweaking such as in Figure 17-right where wrinkles have been easily generated by adding a sinusoid function to the profile-curve. Figure 18 illustrates the tricky example of a slightly soft hat pressing a softer egg-like face. We slightly increased the volume compensation factor ($\gamma = 1.2$) of the latter to exaggerate the bulge effect. The accompanying video shows a transition from a fully rigid hat to a fully rigid egg. Last but not least, Figure 21 shows how tweaking the mapping direction permits to reproduce some friction-like effects.

8.3 Corner cases & limitations

Thin structures. Figure 19 shows a stress test for our method on a sphere-like shape made of thin tubular structures colliding with an equally stiff plane. As can be seen on the left, our method produces a convincing deformation even for the tubes that are deeply immersed in the plane. However, since our method does not use any volumetric structure, such as a tetrahedral mesh, the tubes gets significantly distorted when the collision goes deeper (as seen on the right) until they eventually collapse. For this specific example, a workaround would be to embed the tubular-sphere within a smooth genus-0 bounding triangular mesh (with bounded minimal curvature) onto which our algorithm would be applied, prior to transferring the deformation of this proxy to the detailed tubular mesh.

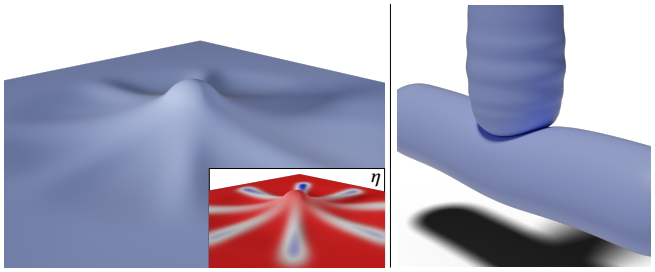


Fig. 17. **Artistic controls.** Left: a sphere is pushing below a plane and a fabric-like effect is achieved through a painted bulge map η with values ranging from one (the default in red), to slightly negative values (in blue). Here we see the *backface* of the planar grid. Right: wrinkles generated by adding a sinusoid to the profile curve. [$\bar{k} = 0.5$, $\gamma = 0.4$, $\tau = 0$]

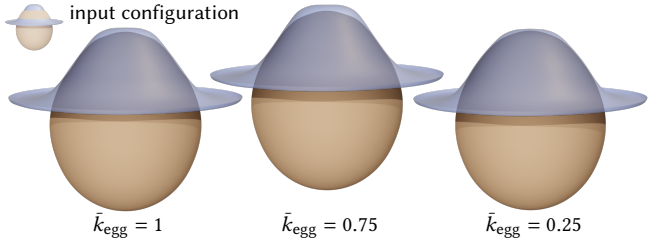


Fig. 18. **Stiffness transition.** From left to right, the relative stiffness is transferred from the hat to the egg. We see the *backface* of the hat surface. [$\tau = 0.2$, $\gamma_{\text{hat}} = 0.3$, $\gamma_{\text{egg}} = 1.2$]

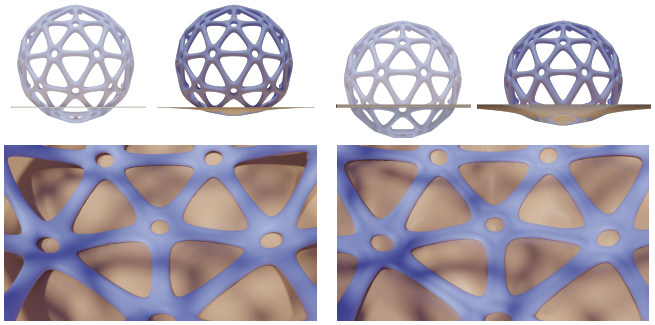


Fig. 19. **Thin structures.** An elastic tubular sphere collides with an elastic plane for two penetration depths (left and right columns). In both cases, our method produces plausible deformations of the overall shapes (first row), but for the deepest collision the tubes start to collapse (bottom right). [$\bar{k}_{\text{sphere}} = 0.56$, $\gamma_{\text{plane}} = 0$, $\gamma_{\text{sphere}} = 1$, $\tau = 0$, $\bar{\phi}_{\text{plane}} = 20$, $\bar{\phi}_{\text{sphere}} = 10$].

Folds. Figure 20 shows another stress test consisting in a sphere colliding with many bended thorns lying on a plane. This example is particularly challenging for our mapping algorithm (Section 4.2) as it creates many complex back/front facing configurations along the mapping direction. Nevertheless, for most parts, the produced deformation remains plausible with some thorns being completely flattened, hence producing expected folds. This example reveals some limitations though. First, since the spikes which are away from the contact zone are deformed along the surface normal, they will unnaturally inflate or shrink when the volume gets redistributed

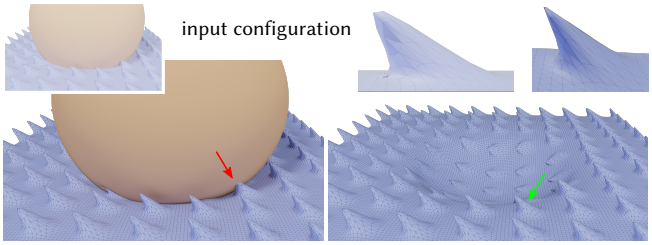


Fig. 20. **Folds.** An elastic sphere collides with many bended elastic thorns. Top-right: close-ups of the thorn highlighted by the green arrow before and after deformation. Bottom-right: same as bottom left but with the sphere hidden to reveal the expected folds. The red-arrow highlights a missed collision. [$\bar{k}_{\text{sphere}} = 0.8$, $\gamma_{\text{sphere}} = 1$, $\gamma_{\text{plane}} = 0$, $\tau = -0.8$, $\bar{\phi}_{\text{sphere}} = 20$, $\bar{\phi}_{\text{plane}} = 4$]

(top right inset in Figure 20). A simple workaround would be to guide the displacement direction along a smoothed normal field. Second, we can see that some collisions between the spheres and the thorns are missed due to the global threshold used by our purely surfacic contact criterion (Section 4.4). This contrasts with the method of Brunel et al. which identifies the contact zone through a volumetric ball-testing procedure that better accounts for the spatial relationships and local variations. Designing a new heuristic combining the best of both approaches makes an interesting future work.

Third, this example recalls that both methods can miss or anticipate local collision events when a new collision occurs nearby a bulge or a depression produced by a previous collision. The many colliding thin thorns exacerbate this issue to the point of producing some flickering during the animation, as shown in the accompanying video. For less challenging configurations, such as the examples of Figure 14, those theoretical discontinuities are not noticeable. This is thanks to both the locality of our fields and the localization term of the anisotropic bulging weights. For the simpler case of two successive collision events, the second one occurring on a previous prominent bulge or depression, a practical workaround would be to apply sequentially our deformer twice, starting with the deepest collision, instead of processing all the components at once. Designing a more general solution to this problem is left for future work.

8.4 Performance

We measured the performance of our prototype implementation on a single core of an Intel i7-4790K CPU. It has been implemented in C++ using Embree in *robust* mode to compute segment-triangle intersections, and the simplicial Cholesky solver of Eigen [Guennebaud et al. 2010] for the different Poisson-like problems. Average breakdown timings are reported for several examples in Table 1. To evaluate the scaling of our method with respect to the size of the input meshes, we deliberately included over-tessellated meshes such as the jelly of Figure 16, the planar ground of Figure 1(a-c), or the simple shapes of Figure 18. Unsurprisingly, computation time is dominated by the ray-triangle intersections and the three matrix factorizations, two for the heat-method and one for the diffusion of the slope and tangent parameters. Overall, the factorizations themselves represent about 25% to 60% of the overall computation. We

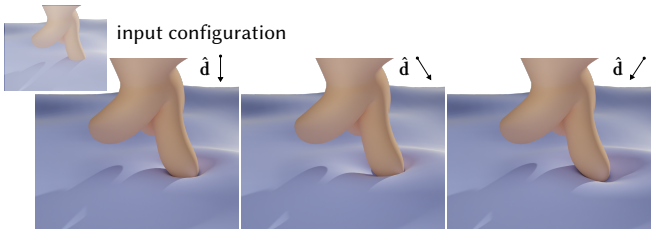


Fig. 21. **Control over the mapping direction $\hat{\mathbf{d}}$.** Left: using the automatic direction. Middle and Right: modifying the mapping direction permits to mimic friction effects. [$\bar{k}_{\text{finger}} = 0.9$, $\bar{\phi}_{\text{ground}} = 50$, $\bar{\phi}_{\text{finger}} = 20$, $\tau = 0$]

Table 1. **Runtime statistics** for pairs of working regions with various number of vertices; average computation time over an animation broken down in terms of detecting the regions in intersection (i.), building the mapping between them (m.) §4.2, extracting the contact zone (c.) §4.4, computing the deformation region parametrization (p.) §5.1, computing the directions of deformation (di.) §5.2, diffusing the guiding fields (g.) §5.3, and deforming the surface according to the profile (de.) §5.3, in percentage of the per frame average total computation time (average). The column (s.) reports the percentage of time spent in matrix factorizations.

scene	$\lvert\mathcal{W}_1\rvert + \lvert\mathcal{W}_2\rvert$ (# vert)	relative time (%)								average time (ms)
		i.	m.	c.	p.	di.	g.	de.	s.	
Fingers-Ball	1064 + 1561	11	9	1	26	4	30	18	33	5.6
Fingers-Palm	2664 + 2184	23	9	1	23	3	21	20	26	9
Skinned finger	1967 + 2951	28	9	1	26	2	20	14	27	13.6
Mouth, fig.1(d)	1825 + 1500	25	13	3	24	1	18	15	26	9.8
Bunny-Jelly	5002 + 14408	12	9	2	43	1	18	16	41	101.8
Walk, fig.1	740 + 11717	5	4	0	30	1	49	11	61	76.8
Hat-Egg	10242 + 12092	41	12	2	30	1	10	5	33	173.4

believe their cost could be significantly cut down by adapting prefactorization techniques [Herholz and Alexa 2018; Herholz et al. 2017]. Intersections could also be sped up using state-of-the-art techniques such as joint BVH traversal and batch intersection tests instead of testing each segment one at a time in random order as currently done in our prototype. Last but not least, vectorization is also a promising approach to further accelerate the overall algorithm.

9 DISCUSSIONS AND CONCLUSION

In this work, we have presented a new deformation technique for resolving local elastic-elastic collisions while producing plausible bulge effects and providing extended artistic controls with instant feedback. It does not aim at generating large, global deformations that should already be handled by skinning, blend-shapes, or cage-based deformers. For example, it should not be used to resolve the collision of the whole, two-sided ear of the Bunny with its back.

We also showed how to extend our pipeline to handle local self-intersections, hence enabling geometric skinning with contact handling, bulge control, and other artistically-driven effects. However, skinning techniques and their associated rigging parameters are designed to prevent such self-intersections by making the deformation unrealistically soft. To exploit our deformer at its full potential, it would be relevant to investigate novel skinning and rigging techniques focusing on the generation of nice deformations on the exterior of the joint while deliberately producing self-intersections on the interior.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their comments. This work has been supported by the ANR *Fold-Dyn* project (ANR-16-CE33-0015). The squid model is part of SOFA. The Stanford Bunny was 3D scanned by the Stanford Computer Graphics Laboratory. *Emily*'s face model is part of the Wikihuman Project.

REFERENCES

Nadine Abu Rumman and Marco Fratarcangeli. 2015. Position-Based Skinning for Soft Articulated Characters. *Computer Graphics Forum* 34, 6 (2015), 240–250.

Alexis Angelidis, Marie-Paule Cani, Geoff Wyvill, and Scott King. 2006. Swirling-Sweepers: Constant-Volume Modeling. *Graph. Models* 68, 4 (2006), 324–332.

Alexis Angelidis and Karan Singh. 2007. Kinodynamic Skinning Using Volume-preserving Deformations. In *Proc. of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 129–140.

Jan Bender, Matthias Müller, and Miles Macklin. 2015. Position-Based Simulation Methods in Computer Graphics. In *EG 2015 - Tutorials*. The Eurographics Association.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Trans. Graph.* 33, 4 (2014), 154:1–154:11.

Yuri Boykov and Vladimir Kolmogorov. 2001. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. In *Proceedings of the Third International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer-Verlag, 359–374.

Camille Brunel, Pierre Bénard, Gaël Guennebaud, and Pascal Barla. 2020. A Time-Independent Deformer for Elastic-Rigid Contacts. *Proc. ACM Comput. Graph. Interact. Tech.* 3, 1 (2020).

Keenan Crane, Clarisse Weischedel, and Max Wardetzky. 2013. Geodesics in Heat: A New Approach to Computing Distance Based on Heat Flow. *ACM Trans. Graph.* 32, 5 (2013), 152:1–152:11.

Crispin Deul and Jan Bender. 2013. Physically-Based Character Skinning. In *Virtual Reality Interactions and Physical Simulations (VRIPhys)*. Eurographics Association.

Ming Gao, Nathan Mitchell, and Eftychios Sifakis. 2014. Steklov-Poincaré Skinning. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 139–148.

J.-P. Gourret, N. M. Thalmann, and D. Thalmann. 1989. Simulation of Object and Human Skin Formations in a Grasping Task. *SIGGRAPH Comput. Graph.* 23, 3 (1989), 21–30.

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

David Harmon, Daniele Panozzo, Olga Sorkine, and Denis Zorin. 2011. Interference-aware Geometric Modeling. *ACM Trans. Graph.* 30, 6 (2011), 137:1–137:10.

Philipp Herholz and Marc Alexa. 2018. Factor Once: Reusing Cholesky Factorizations on Sub-Meshes. *ACM Transaction on Graphics* 37, 6 (2018).

Philipp Herholz, Timothy A. Davis, and Marc Alexa. 2017. Localized solutions of sparse linear systems for geometry processing. *ACM Transactions on Graphics* 36, 6 (2017).

Aaron Hertzmann and Denis Zorin. 2000. Illustrating Smooth Surfaces. In *Proc. of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. 517–526.

Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. 2008. Geometric Skinning with Approximate Dual Quaternion Blending. *ACM Trans. Graph.* 27, 4 (2008), 105:1–105:23.

Ladislav Kavan and Olga Sorkine. 2012. Elasticity-inspired Deformers for Character Articulation. *ACM Trans. Graph.* 31, 6 (2012), 196:1–196:8.

Martin Komaritzan and Mario Botsch. 2018. Projective Skinning. *Proc. ACM Comput. Graph. Interact. Tech.* 1, 1 (2018), 12:1–12:19.

John Lasseter. 1987. Principles of Traditional Animation Applied to 3D Computer Animation. *SIGGRAPH Comput. Graph.* 21, 4 (1987), 35–44.

Binh Huy Le and Jessica K. Hodgins. 2016. Real-time Skeletal Skinning with Optimized Centers of Rotation. *ACM Trans. Graph.* 35, 4 (2016), 37:1–37:10.

Binh Huy Le and J P Lewis. 2019. Direct Delta Mush Skinning and Variants. *ACM Trans. Graph.* 38, 4 (2019), 113:1–113:13.

J. P. Lewis, Matt Cordner, and Nickson Fong. 2000. Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton-driven Deformation. In *Proc. of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 165–172.

Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M. Kaufman. 2020. Incremental Potential Contact: Intersection- and Inversion-free Large Deformation Dynamics. *ACM Transactions on Graphics* 39, 4 (2020).

Yijing Li and Jernej Barbič. 2019. Multi-Resolution Modeling of Shapes in Contact. *Proc. ACM Comput. Graph. and Interact. Tech.* 2, 2 (2019).

Libin Liu, KangKang Yin, Bin Wang, and Baining Guo. 2013. Simulation and Control of Skeleton-driven Soft Body Characters. *ACM Trans. Graph.* 32, 6 (2013), 215:1–215:8.

N. Magnenat-Thalmann, R. Laperrière, and D. Thalmann. 1988. Joint-dependent Local Deformations for Hand Animation and Object Grasping. In *Proceedings on Graphics Interface '88*. Canadian Information Processing Society, 26–33.

Aleka McAdams, Yongning Zhu, Andrew Selle, Mark Empey, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. 2011. Efficient Elasticity for Character Skinning with Contact and Collisions. *ACM Trans. Graph.* 30, 4 (2011), 37:1–37:12.

Andrew Nealen, Matthias Müller, Richard Keiser, Eddy Boxerman, and Mark Carlson. 2006. Physically Based Deformable Models in Computer Graphics. *Computer Graphics Forum* 25, 4 (2006), 809–836.

Jesús R. Nieto and Antonio Susín. 2013. *Cage Based Deformations: A Survey*. Springer, 75–99.

Mark Pauly, Dinesh K. Pai, and Leonidas J. Guibas. 2004. Quasi-Rigid Objects in Contact. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 109–119.

Damien Rohmer, Stefanie Hahmann, and Marie-Paule Cani. 2009. Exact Volume Preserving Skinning with Shape Control. In *Proc. of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 83–92.

Thomas W. Sederberg and Scott R. Parry. 1986. Free-form Deformation of Solid Geometric Models. *SIGGRAPH Comput. Graph.* 20, 4 (1986), 151–160.

Breannan Smith, Fernando de Goes, and Theodore Kim. 2018. Stable Neo-Hookean Flesh Simulation. *ACM Trans. Graph.* 37, 2 (2018), 12:1–12:15.

Olga Sorkine and Marc Alexa. 2007. As-Rigid-As-Possible Surface Modeling. In *Proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*. 109–116.

Olga Sorkine and Mario Botsch. 2009. Interactive Shape Modeling and Deformation. In *Eurographics 2009 - Tutorials*, K. Museth and D. Weiskopf (Eds.). The Eurographics Association.

Yun Teng, Miguel A. Otaduy, and Theodore Kim. 2014. Simulating Articulated Subspace Self-contact. *ACM Trans. Graph.* 33, 4 (2014), 106:1–106:9.

Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. 1987. Elastically Deformable Models. *SIGGRAPH Comput. Graph.* 21, 4 (1987), 205–214.

Rodolphe Vaillant, Loïc Barthe, Gaël Guennebaud, Marie-Paule Cani, Damien Rohmer, Brian Wyvill, Olivier Gourmel, and Mathias Paulin. 2013. Implicit Skinning: Real-time Skin Deformation with Contact Modeling. *ACM Trans. Graph.* 32, 4 (2013), 125:1–125:12.

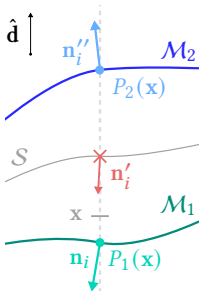
Rodolphe Vaillant, Gäel Guennebaud, Loïc Barthe, Brian Wyvill, and Marie-Paule Cani. 2014. Robust Iso-surface Tracking for Interactive Character Skinning. *ACM Trans. Graph.* 33, 6 (2014), 189:1–189:11.

Wolfram von Funck, Holger Theisel, and Helmut Seidel. 2008. Volume-preserving Mesh Skinning. In *Proc. of the Vision, Modeling, and Visualization Conference*.

Wolfram von Funck, Holger Theisel, and Hans-Peter Seidel. 2006. Vector Field Based Shape Deformations. *ACM Trans. Graph.* 25, 3 (2006), 1118–1125.

Jiayi Eris Zhang, Seungbae Bang, David I. W. Levin, and Alec Jacobson. 2020. Complementary Dynamics. *ACM Trans. Graph.* 39, 6 (2020).

A POTENTIAL SURFACE NORMALS COMPUTATION



For the sake of symmetry, to compute the normal of each vertex projected on $\mathcal{S}$, we use an implicit surface representation of $\mathcal{S}$. We derive its normals by calculating the gradient of the associated scalar field at this position. Let us consider a point $\mathbf{x}$. $\mathrm{P}_1(\mathbf{x})$ and $\mathrm{P}_2(\mathbf{x})$ are its associated projection points along $\hat{\mathbf{d}}$ on $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively. We want to find the gradient on $\mathcal{S}$ at $\mathbf{x}$. By definition, this point is at the intersection of $\mathcal{S}$ and the segment joining $\mathrm{P}_1(\mathbf{x})$ and $\mathrm{P}_2(\mathbf{x})$. We thus define the following scalar field:

$$F(\mathbf{x}) = ((\bar{k}_2\,\mathrm{P}_1(\mathbf{x}) + (1-\bar{k}_2)\,\mathrm{P}_2(\mathbf{x})) - \mathbf{x})^\top \cdot \hat{\mathbf{d}}$$

The zero iso-surface $F = 0$ defines the implicit surface corresponding to $\mathcal{S}$. To obtain the normal at each point of this implicit surface, we now need to compute the gradient of this function:

$$\nabla F = \bar{k}_2\,\nabla(\mathrm{P}_1(\mathbf{x})^\top \cdot \hat{\mathbf{d}}) + (1-\bar{k}_2)\,\nabla(\mathrm{P}_2(\mathbf{x})^\top \cdot \hat{\mathbf{d}}) - \hat{\mathbf{d}}.$$

Taking the linear approximation of P_1 and P_2, we finally get:

$$\nabla F = (I - \hat{\mathbf{d}}^\top\hat{\mathbf{d}})\left(\frac{\bar{k}_2}{\mathbf{n}_1^\top\hat{\mathbf{d}}}\,\mathbf{n}_1 + \frac{(1-\bar{k}_2)}{\mathbf{n}_2^\top\hat{\mathbf{d}}}\,\mathbf{n}_2\right) - \hat{\mathbf{d}},$$

with $\mathbf{n}_1$ the normal on $\mathcal{M}_1$ at this point, and $\mathbf{n}_2$ the interpolated normal on $\mathcal{M}_2$ at $\mathrm{P}_2(\mathbf{x})$. To avoid numerical instabilities when normalizing the vector, we eliminate the denominators in the previous equation to obtain the final direction associated to $\mathcal{M}_1$:

$$\eta_{i_1} = (I - \hat{\mathbf{d}}^\top \hat{\mathbf{d}}) \left(\frac{\bar{k}_2 \, |\mathbf{n}_{i_2}^\top \hat{\mathbf{d}}|}{\mathrm{sign}(\mathbf{n}_{i_1}^\top \hat{\mathbf{d}})} \, \mathbf{n}_{i_1} + \frac{(1-\bar{k}_2) \, |\mathbf{n}_{i_1}^\top \hat{\mathbf{d}}|}{\mathrm{sign}(\mathbf{n}_{i_2}^\top \hat{\mathbf{d}})} \, \mathbf{n}_{i_2} \right) - |\mathbf{n}_{i_1}^\top \hat{\mathbf{d}}| \, |\mathbf{n}_{i_2}^\top \hat{\mathbf{d}}| \, \hat{\mathbf{d}},$$

The final normal of vertex i of $\mathcal{M}_1$ on $\mathcal{S}$ is then:

$$\mathbf{n}'_{i_1} = \frac{\eta_{i_1}}{\|\eta_{i_1}\|}.$$

To obtain the final normal of a vertex j of $\mathcal{M}_2$ on $\mathcal{S}$, we replace $\hat{\mathbf{d}}$ by $-\hat{\mathbf{d}}$ in previous equations.

B GENERAL ALGORITHM

Algorithm Part 1 - Contact zone

Input: a set of triangle mesh working regions $\{\mathcal{W}_l\}$
Output: the deformed elastic surface positions $\mathbf{p}'$

for all pairs $(\mathcal{W}_a, \mathcal{W}_b)$ in intersection **do**
  Intersection detection ▷ [Brunel et al. 2020]
    Compute and store edge-face intersection points, yielding intersection regions $\mathcal{I}_a$, $\mathcal{I}_b$
  Mapping direction ▷ §4.1
    If not provided, estimate $\hat{\mathbf{d}}_{a,b}$ through integrals over $\mathcal{I}_a$ and $\mathcal{I}_b$ using Eq. 2 ($\hat{\mathbf{d}}_{b,a} \leftarrow -\hat{\mathbf{d}}_{a,b}$)
  Region mapping ▷ §4.2
    for all $(a', b') \in \{(a,b),(b,a)\}$ **do**
      $Q \leftarrow \{\}$ ▷ set of extruded contour quads
      for all face f in $\mathcal{I}_{a'}$ exhibiting a contour segment s **do**
        $Q \leftarrow Q \cup \{\text{extruded\_quad}(s, \hat{\mathbf{d}}_{a',b'})\}$
      for all front facing edge ij of $\mathcal{W}_{b'}$ intersecting Q **do**
        Tag its extremities as either *inside* or *outside*
      Propagate the *inside* tag yielding $\mathrm{P}_{b'}(\mathcal{B}_{a'})$
      $\mathcal{M}_{b'} \leftarrow \mathrm{P}_{b'}(\mathcal{B}_{a'}) \cup \mathcal{B}_{b'}$
      for all vertex $\mathbf{p}_i$ in $\mathcal{M}_{b'}$ **do**
        $\mathbf{p}''_i \leftarrow \text{farthest\_intersection}(\mathcal{W}_{a'}, \text{ray}(\mathbf{p}_i, \hat{\mathbf{d}}_{a',b'}))$
  Potential contact surface $\mathcal{S}$ ▷ §4.3
    for all $(a', b') \in \{(a,b),(b,a)\}$ **do**
      for all vertex i in $\mathcal{M}_{a'}$ **do**
        $\mathbf{p}'_i \leftarrow \mathbf{p}_i + (1 - \frac{k'_a}{k_a + k_b})(\mathbf{p}''_i - \mathbf{p}_i)$
  Contact zone extraction ▷ §4.4
    Compute the threshold ε_c using Eq. 4 and integrals over $\mathcal{S}$
    for all $l \in \{a, b\}$ **do**
      $C_l \leftarrow \{\mathbf{p}_i, i \in \mathcal{M}_l \mid \mathbf{p}_i \text{ satisfies Eq. 3 }\}$
      for all edge ij with $i \in C_l$, and $j \notin C_l$ **do**
        Find α_{ij} using Eq. 5
        $\partial C_l \leftarrow \partial C_l \cup \{(1-\alpha_{ij})\,\mathbf{p}_i + \alpha_{ij}\,\mathbf{p}_j\}$
    if C_a and C_b are not empty **then**
      $R_a \leftarrow R_a \cup \{b\}$; $R_b \leftarrow R_b \cup \{a\}$ ▷ sets of contact zone indices
end ▷ continues in Part 2

Algorithm Part 2 - Deformation

for all $\mathcal{W}_l$ **do**
  Deformable region parametrization ▷ §5.1
    Solve $(\mathrm{id} - t\Delta)v = 0$ in a least-squares sense with the linear constraints $(1-\alpha_{ij})\,v_i + \alpha_{ij}\,v_j = 1$, $\forall$ edge ij crossing ∂C_l
    Compute normalized gradients $X = -\nabla v / \|\nabla v\|$
    Solve Poisson equation $\Delta\phi = \nabla \cdot X$ ensuring $\phi = 0$ on ∂C_l
    $u \leftarrow \phi / \phi_{\max,l}$ ▷ $\phi_{\max,l}$ is the deformation extent
    $\mathcal{D}_l \leftarrow \{\text{vertex } i \in \mathcal{W}_l \mid u_i \in]0,1]\}$ ▷ deformable region
  Direction field computation ▷ §5.2
    if $|R_l| > 1$ **then**
      Solve $\Delta\bar{\mathbf{d}} = 0$ with Neumann condition on $\partial\mathcal{W}_l$ and $\bar{\mathbf{d}} = \hat{\mathbf{d}}_{l,k}$ on $\partial C_{l,k}$, $\forall k \in R_l$
    else
      $\bar{\mathbf{d}} \leftarrow \hat{\mathbf{d}}_{l,k}$ with $R_l = \{k\}$
    $\mathbf{d}_i \leftarrow \text{normalize}\left(w(u_i)\,\bar{\mathbf{d}}_i + (1 - w(u_i))\,\mathbf{n}_i\right)$, $\forall i \in \mathcal{D}_l$ ▷ Eq.6
  Amplitude a and slope s fields computation ▷ §5.3
    Estimate the amplitude $a_{ij} = \|(1-\alpha_{ij})(\mathbf{p}'_i - \mathbf{p}_i) + \alpha_{ij}(\mathbf{p}'_j - \mathbf{p}_j)\|$, $\forall$ edge ij crossing ∂C_l
    Interpolate a over $\mathcal{D}_l$ by harmonic diffusion with:
      ▷ $a_j = \sum_i \alpha_{ij}\, a_{ij} / \sum_i \alpha_{ij}$, $\forall$ edge ij crossing ∂C_l
      ▷ Natural Neumann condition on the rest of $\partial\mathcal{D}_l$
    Estimate the slopes s_{ij}, $\forall$ edge ij crossing ∂C_l
    Interpolate s over $\mathcal{D}_l$ by harmonic diffusion with:
      ▷ $s_j = \sum_i \alpha_{ij}\, s_{ij} / \sum_i \alpha_{ij}$, $\forall$ edge ij crossing ∂C_l
      ▷ Natural Neumann condition on the rest of $\partial\mathcal{D}_l$
  Profile curves instantiation ▷ §5.3
    Pre-compute the first two control points defining the 2D parametric B-Spline $(f_1(t), f_2(t)) = \sum_k N_k^3(t)\mathbf{v}_k$:
    for all vertex $i \in \mathcal{D}_l$ **do**
      $\theta_i \leftarrow \tan^{-1}(s_i / a_i)$
      $\mathbf{v}_0 \leftarrow (0, -a_i)$
      $\mathbf{v}_1 \leftarrow (0.2\cos\theta_i,\ a_i(0.4\sin\theta_i - 1))$
      $t_i \leftarrow f_1^{-1}(u_i)$
      Compute η_i ▷ Eq.7
    Compute h_v to preserve the volume exactly ▷ degree 3 equation
    for all vertex $i \in \mathcal{D}_l$ **do**
      Set the ordinate of $\mathbf{v}_2$ as $\gamma \cdot \eta_i \cdot h_v$
      $\mathbf{p}'_i \leftarrow \mathbf{p}_i + f_2(t_i)\,\mathbf{d}_i$ ▷ displacement
end